Challenges with Learning to Program and Problem Solve: An Analysis of Student Online Discussions

Conference or Workshop Item

Challenges with Learning to Program and Problem Solve: An Analysis of Student Online Discussions

Paul Piwek*
Simon Savage*
Paul.Piwek@open.ac.uk
S.A.Savage@open.ac.uk
School of Computing and Communications
The Open University
Milton Keynes, United Kingdom

ABSTRACT

Students who study problem solving and programming (in a language such as Python) at University level encounter a range of challenges, from low-level issues with code that won't compile to misconceptions about the threshold concepts and skills. The current study complements existing findings on errors, misconceptions, difficulties and challenges obtained from students after-the-fact through instruments such as questionnaires and interviews. In our study, we analysed the posts from students of a large cohort (~1500) of first-year University distance learning students to an online 'Python help forum' - recording issues and discussions as the students encountered specific challenges. Posts were coded in terms of topics, and subsequently thematically grouped into Python-related, problem solving/generic programming related, and module specific. We discuss the set of topics and rank these in terms of the number of forum discussions in which they occur (as a proxy for their prevalence). The top challenges we identified concern student understanding and use of a mix of programming environments (in particular, Python IDLE for offline programming and CodeRunner for programming quizzes) and code fragment problems. Apart from these, Python-specific topics include, among others, collections, functions, error messages, iteration, outputting results, indentation, variables and imports. We believe that the results provide a good insight into the challenges that students encounter *as they learn to program.* In future work we intend to study the discussions in further detail in terms of theories of conceptual change.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; *Computational thinking*; *Software engineering education*; *Adult education*; • **Applied computing** → **Distance learning**; **E-learning**.

*Both authors contributed equally to this research.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366838

KEYWORDS

programming, Python, problem solving, online student discussions, challenges, misconceptions, threshold concepts and skills

ACM Reference Format:
Paul Piwek and Simon Savage. 2020. Challenges with Learning to Program and Problem Solve: An Analysis of Student Online Discussions. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3328778.3366838

1 INTRODUCTION

The introduction to a text-based procedural programming language poses a range of challenges for students, especially when this is done in a distance learning environment. Introducing students to programming (and problem solving) is generally acknowledged as difficult or at least perceived as a significant challenge [4, 5].

Students encounter threshold concepts and skills [9, 14, 15] and can get entangled in misconceptions [3, 12]. These issues may be amplified in distance and, more specifically, online learning environments, which have gained ground not only in the form of MOOCs and courses offered by online education providers, but also in more traditional teaching establishments that apply the 'flipped classroom' approach [1].

The purpose of the current study is to examine the challenges that first year distance learning students face when learning a procedural programming language such as Python over the course of a 21-week computing and IT course which includes 6 weeks on problem solving and programming with Python. So far, much of the evidence on students' challenges is based on retrospective surveys/interviews with students and instructors, and analysis of student assessment results or student-authored concept maps [12, 15]. To our knowledge, no results draw on large data sets of contemporary student discussions in their own language around challenges. The current study provides an additional perspective by examining the evidence from a large collection of student online discussions as they learn to program.

In Section 2, we summarise research methods and findings from the literature on the challenges faced by students when learning to program. Section 3 introduces the methodology of the current study, which uses (online) discussions that took place as students encountered and struggled with specific challenges. Section 4 presents the results of our study, grouping challenges by topic and, at a higher level, into themes. In Section 5 we discuss our findings. Finally,

our conclusions and suggestions for further work are presented in Section 6.

2 RELATED WORK

A wide range of instruments have been used to study the challenges that students face when learning to program. Perhaps the most direct evidence comes from studies that analyse compiler error messages when students attempt to execute their code. [8] found that such errors (in Java) can be classified into different types with good reliability. Errors found this way can be ordered by frequency and therefore provide some indication of their prevalence. However, the error types are relatively low-level – e.g. the top five reported by [8] are: variable not declared, colon missing, incorrectly written variable name, invalid syntax, method naming incorrect.

In terms of [12] most of these errors go back to difficulties in syntactic knowledge. However, [12] identify two further levels at which misconceptions can arise: the conceptual and strategic levels. E.g., at the conceptual level a student may fail to understand that a variable can only hold one value at a time. Strategic knowledge or know how concerns expertise on how to go from a problem to an implemented solution for that problem. This includes for instance problem decomposition, development of an algorithm and implementation of this algorithm in a specific language, whilst testing and debugging the code as it is developed.

Within computing, specific attention has been given to 'Threshold concepts', originally proposed by [9] as transformative, integrative, irreversible, potentially troublesome and often indicating the boundaries of a discipline. Early research identified for instance objected orientation and pointers as threshold concepts [2]. More recently, [15], in their literature review, enumerate a long list of further concepts that have since been identified, from data abstraction and design patterns to polymorphism and program-memory interaction. Additionally, [14] highlight that not only concepts, but also skills, can play the role of thresholds in computing.

[15] enumerate the instruments that have been used so far for collecting data on threshold concepts. They distinguish evidence from faculty and students. Much of the evidence focuses on after-the-fact data collection through surveys and interviews. Similarly, work building concept inventories (e.g. [3], a concept inventory is a multiple-choice questionnaire where each distractor answer maps to a specific student misconception) typically relies on questionnaires to gather initial evidence on where students face difficulties in understanding concepts.

[17] argue that collecting evidence from students on where they experienced difficulties in the past can be unreliable, with students struggling to recall and recount specific occasions on which challenges were encountered. This suggest that records of challenges as they occur may be more informative on actual problems that students encounter.

As described by [10], specific challenges can arise in distance learning contexts. These are related to the type of students (distance learning students often study part-time whilst being in full-time employment) and reliance on technology to allow students to communicate and collaborate with each other.

3 METHODOLOGY

Our research question is: *What are the challenges that first-year distance learning students face when learning to program in Python?* Rather than rely on post-hoc accounts of what students remember about the challenges they encountered, we examine records of discussions around the challenges *as they emerged* for our students.

The results in this study concern the cohort that took *Introduction to computing and information technology 2* (henceforth TM112) at the UK's Open University from April to September 2018. A further detailed description of the module is provided elsewhere (see [11]).

The TM112 module included a dedicated online 'Python help forum' where students could discuss challenges with their study of Python. The online forum relied primarily on peer support (enabled partly by a broad range of backgrounds among our students, from absolute novices to programming experts seeking a qualification). We also observed that students were particularly effective in providing emotional support and motivating each other, when they faced difficulties. In addition to the peer support, there was a team of moderators (three) who monitored the forum discussions and would steer the discussion if needed.

3.1 Student demographics

We studied the 2018 TM112 cohort of about 1500 students. More than half of students were between 25 and 39 years old; 10 students were over 65 years old; approximately 200 were under 21. The female to male ratio is 24 to 76. Race/ethnicity of students (rounded): 89% of students were White, 2% Black, 3% Asian, 1% Mixed, 1% Other and 3% did not specify. 19% of students declared a disability. 16% of students are classified as low socio-economic status by the university, with a further 5% being unknown.

TM112 is the second module for most Computing and Information Technology undergraduate students at the Open University and is also taken by some students on other pathways, such as Data Science. Most students will have completed a predecessor module *Introduction to computing and information technology 1* (TM111) [18] which introduces students to University level study, a range of computing and IT topics and basic programming in a visual programming language.

All students (both those who completed and those who did not complete the module) are surveyed at the end of the module (before they receive their results). The response rate for the survey for the April 2018 cohort was 16%. Most students, over 90% agreed with the statement 'I was satisfied with the quality of the module' (and less than 5% disagreed). About 80% agreed that 'My studies have helped me develop my self-confidence', (less than 4% disagreed). Also, more than 90% agreed that 'It was obvious how the module materials related to the assessed tasks on this module', (less than 3% disagreed).

3.2 Instructor demographics

The online 'Python help' forum was moderated by two tutors (male and female). Additionally, the course leader (male) monitored the forums and supported the two tutors. All three staff involved are white and have, each individually, over 20-years of computing and Information Technology teaching experience.

3.3 Programme components

TM112 runs over 21 study weeks (300 hours in total for 30 study credits). Six of these weeks are on 'Problem solving with Python':

Week 2 Sequence, selection, variables, lists and (nested) iteration, Python Turtles library.

Week 4 Formula problems, case analysis, Booleans, testing, documentation, pattern for generating a sequence.

Week 7 Generating lists, Reduce (count and aggregate), Search (finding a value/the best value), Combining patterns.

Week 9 Python functions (and automated testing of functions with assert), Python objects and names.

Week 10 Worked example of analysis, with Python, of Office for National Statistics (ONS) health and wellbeing dataset.

Week 15 Worked example implementing a simple flashcards program for the module glossary using Python dictionairies, interactive loops, the random library and reading from a file.

During each week of TM112, students work with printed materials and online activities and during some weeks attend tutorial events (face-to-face or online). At the end of each week, there is a formative quiz. To encourage students to engage with the quizzes, they are rewarded with a small number of marks for including evidence of engagement with the questions in their assignments. Marks are for the evidence of engagement and personal narrative/reflection on their engagement with the quiz questions. Since the quiz questions were not summatively assessed, students were also encouraged to discuss their attempts and answers with their peers on the module forums. For the weeks covering Python, see above, they are referred specifically to the Python help forum.

Students were made aware of the forums through several routes. Initial contact from their tutors mentioned the forums and that it is permissible to discuss quiz questions in them. This latter point was reinforced in the introductory video that students watch during their first week of study. Students were also encouraged to explore the module website, including the forums, during that first week and the 'Python Help' contained a pinned post explaining that it is okay to discuss quiz questions in it. Students who posted in less appropriate forums were redirected to the correct forum. Forums consist of a series of discussions, with each discussion consisting of one or more posts.

3.4 Data analysis methods

Our raw data was the full collection of posts to the Python help forum. We extracted the post data from the forum site and stored it in a spreadsheet. Information which could identify participants was anonymised. One or more topics were assigned to each post, providing a descriptive label for the content of the post (i.e. during the First Cycle coding we applied Descriptive coding [13]). In the first instance, all coding was carried out by one of the authors and subsequently validated by the other author. On some occasions, some topics were combined. Coding at the level of individual posts means that some discussions encompassed multiple topics. A second thematic grouping of the topic labels was carried out (as part of the Second Cycle coding [13]). This led us to a division of topics into three higher level themes: Python related, problem solving/generic programming related, and module specific. The top ten of topics that were identified for the first two themes are shown in Tables 1 and 2 together with quantitative information that emerged based on the qualitative coding.

3.5 Scope and limitations

There is a limitation with the core data in that it doesn't record the engagement of non-speakers. This means that we can't accurately measure how individual discussions reverberated within the student community. We can, however, be confident that discussions are being read with some potentially attracting hundreds of readers.

The methodology does have other potential limitations. The manual nature of the coding process means that it may be biased by the opinion of the encoder's choice of a single topic and turn-type per turn. This is somewhat mitigated by team members reviewing the coding process to ensure consensus, although inter-rater reliability scores were not calculated.

Furthermore, the 'Python Help' forum's primary purpose was to support students in their understanding of Python concepts and problem-solving concepts raised in TM112 (see above) with the tools that were available in TM112. Topics which are beyond TM112 are unlikely to be raised, thus skewing the results in favour of TM112-related concepts. Similarly, the student cohort in question was constituted primarily of part-time distance learning undergraduate computing and information technology students, whose profile is likely to differ from, for instance, full-time computer science students. For many of our students, TM112 helps to decide the subsequent study pathway, with options ranging from computer science (with a significant programming and theoretical CS component) to information and communications technology.

The 'Python Help' forum is unlikely to be the students' only point of support and we could reasonably expect them to utilise their tutors, external social network facilities and specialist websites for support. It is not possible to quantify the effect of this on our results.

3.6 Data collection and analysis approval

All data collection and analysis complied with approval processes and methods (regarding both ethics of research with human participants/students and data protection) at the University, ensuring participant privacy, confidentiality, and protection. Neither participants nor researchers received monetary or gift incentives. The data analysis was financially supported through a grant from the UK's Institute of Coding.

4 RESULTS

The Python help forum contained 178 discussions with a total of 1430 posts. The encoding process identified 63 topics within the forum posts: 29 Python-related, 19 on problem solving and general programming skills, and 15 focusing on module-specific questions and issues. Tables 1 and 2 show the top ten Python and problem-solving/generic topics. For the full set of topics, their definitions and example snippets from the forum discussions, see [16].

The IDE was an issue in 40 discussions (22%). Prominent in these was confusion between how to use the IDLE Shell and the Editor. For example, a student experienced a syntax error. It transpired that they had authored their code in the Shell, saved it as a .py file and then tried opening it in the Editor. Apart from the Editor and

Table 1: Statistics for top ten Python-related topics

Topic	Number of discussions	Min. discus. size (posts)	Max. discus. size (posts)	Rounded mean discus. size (posts)	Rounded median discus. size (posts)
IDE	40	1	57	10	7
Collections	21	4	57	13	8
Functions	16	2	31	14	12
Error messages	15	1	57	12	10
Iteration	14	2	39	11	9
Outputting results	12	2	33	14	9
Indentation	10	3	16	8	6
Variables	8	3	57	15	8
Imports	7	7	17	12	11
If structures	5	2	30	11	5

Table 2: Statistics for top ten problem-solving/generic topics

Topic	Number of discussions	Min. discus. size (posts)	Max. discus. size (posts)	Rounded mean discus. size (posts)	Rounded median discus. size (posts)
Code fragment problem	73	1	57	9	6
Bug finding	21	2	20	8	5
Learning Python	15	6	57	16	14
Maths	10	2	57	12	7
Code review	7	7	57	24	21
Problem-solving workflow	6	1	30	17	17
Code explanation	5	5	30	13	8
Following instructions	5	10	18	13	12
Patterns	3	7	12	10	10
Algorithm	2	10	36	23	23

Shell, students also wrote code in the browser with CodeRunner [7]. In particular, CodeRunner was used for the module's formative quizzes involving programming questions. An issue here was students wanting to test their answers in IDLE prior to submitting, a reasonable approach since the quiz penalises incorrect answers. However, they hadn't appreciated that the quiz questions may have underpinning supporting code. Generally, some students found it challenging to fully grasp the purpose and details of authoring code using these three different tools.

For the problem solving/generic topics, in terms of number of discussions, code fragment problems were most prevalent. Typically, these posts initiated a discussion where students knew they needed support, but in many cases the information provided was insufficient to receive targeted support and indicated that they may not have the language, confidence, or knowledge to seek that support. For example:

> 'hello all would be grateful for advice i am practising python anf from the book have just typed in program 2:4 but when running programme it gives me error code saying traceback but have followed completely to process so should be no problems please see diagram'

Unfortunately, the diagram wasn't provided despite requests from other forum contributors. Another student starts with:

> 'Just catching up with the quiz, but can't see where I'm going wrong; am just getting the moderate earthquake response >= line 6 (haha.). Any direction would be great. (...)'

The student who posted this contribution did include, with their posting, a screen capture with part of their code, as shown in Figure 1, and the feedback from the CodeRunner programming environment.

5 DISCUSSION

For reasons of space we have only described first-ranked topics for two themes (with further detail available in [16]) – nevertheless, it is notable that in terms of number of discussions they outnumbered the second-listed items by a factor of 2 and 3, respectively (with frequency of further lower-ranked topics also quickly decreasing).

When we compare our results with those in the literature, we find a significant degree of agreement. For instance, even though [3] focuses on misconception relating to Java, there is a definite similarity with our results: e.g., their Table 4 includes topics such as returning values, calling functions, iteration, conditionals and maths which match with ones found in our study.

Many of our Python specific topics seem to be under 'Basic Programming Principles' in Table 1 of the literature review on threshold concepts by [15]. In our Python-specific top ten, functions

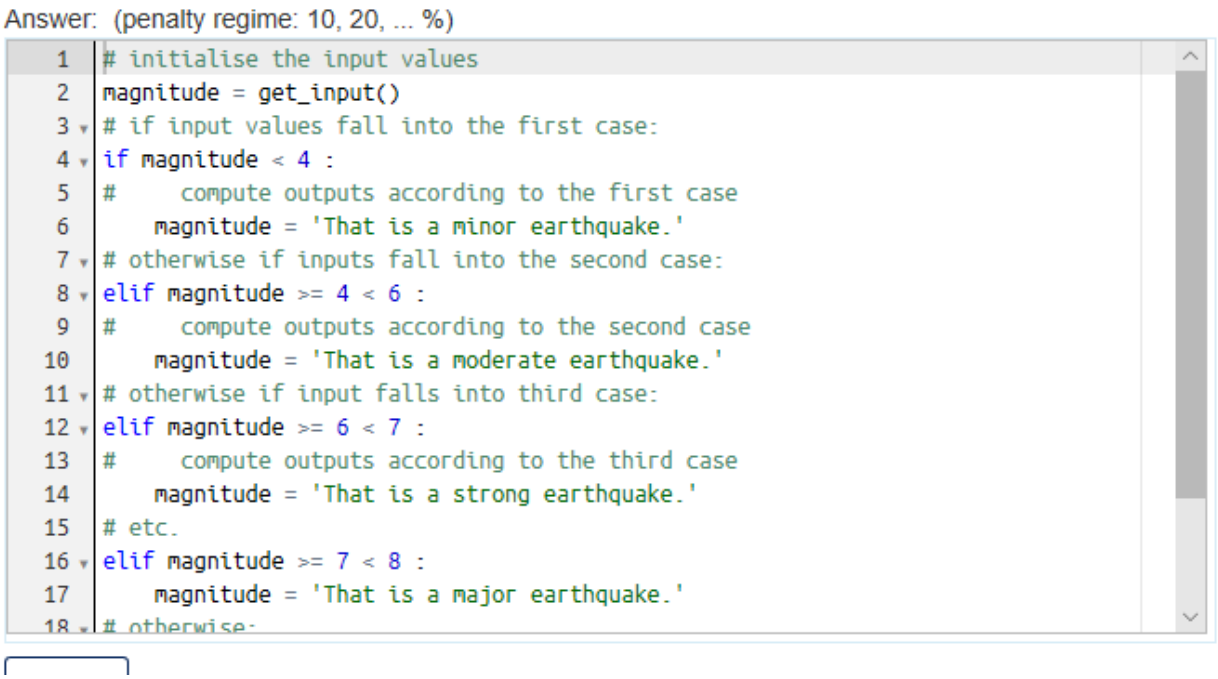

Figure 1: Example of a student forum contribution of a screen capture as part of a code fragment problem

and outputting results resonate with the identification of function-related threshold concepts in [6].

At the thematic level, [12]'s syntactic and conceptual levels (variables, conditional expressions, loops, etc.) correspond with our Python-specific topics. Their strategic level corresponds to our Problem solving/generic topic level.

Despite this significant overlap between our findings and those in the literature, as we already saw, our top-ranked items are not in any of the results reported previously. Of course, our study has certain limitation and a specific scope (as described in Section 3), but nevertheless some preliminary implications can be drawn relating to these top-ranked challenges. Firstly, we would like to highlight that conceptual issues can also emerge in relation to tool use (such as IDEs). Early on in TM112, the focus was mostly on the how, rather than the underlying conceptual understanding of code execution. This has been amended in a subsequent presentation of TM112 with informal observations suggesting a positive effect. A note of caution also emerges from our findings in relation to the recommendation by Qian and Lehman [12] to make more use of existing tools. Our results did suggest that students can find it challenging to cope with several code authoring tools when learning to program.

With regards to problem solving skills, we observed that some students struggle to solicit help, suggesting they need support early on with formulating questions about their own code. For instance, students could be encouraged to always include the code they wrote, expected output and actual output (rather than a general comment stating their code doesn't work).

6 CONCLUSION AND FURTHER WORK

Our methodology, which differs in significant ways from most existing work on challenges for beginning programmers, has resulted in findings that confirm many of the misconceptions and threshold concepts that have been identified. However, the top-ranked challenges in our study are not found in the previous literature. This suggests further investigation, especially into tools used with beginning programmers (and the potential benefits but also downsides) and ways to help students better express problems with their code. We agree with Qian and Lehman [12] that a deeper understanding will require studying the challenges that are posed at a more detailed level in terms of conceptual change theories. In future work, we aim to further explore our data set of forum discussions, performing an in-depth analysis of the learning and conceptual change that occurs in the course of the discussions.

ACKNOWLEDGMENTS

This work has been completed with support from The Institute of Coding, an initiative funded by the UK Office for Students.

REFERENCES

[1] Jacob Lowell Bishop, Matthew A Verleger, et al. 2013. The flipped classroom: A survey of the research. In *ASEE national conference proceedings, Atlanta, GA*, Vol. 30. 1–18.

[2] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, Kate Sanders, and Carol Zander. 2007. Threshold Concepts in Computer Science: Do They Exist and Are They Useful?. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*. ACM, New York, NY, USA, 504–508. https://doi.org/10.1145/1227310.1227482

[3] Ricardo Caceffo, Pablo Frank-Bolton, Renan Souza, and Rodolfo Azevedo. 2019. Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*. ACM, New York, NY, USA, 23–29. https://doi.org/10.1145/3304221.3319771

[4] Chris Dobbyn and Frances Chetwynd. 2014. Transforming retention and progression in a new Level 1 course. Retrieved August 29, 2019 from http://www.open.ac.uk/about/teaching-and-learning/esteem/sites/www.open.ac.uk.about.teaching-and-learning.esteem/files/files/ecms/web-content/2014-05-C-Dobbyn-and-F-Chetwynd-final-report-May-14.pdf eSTEeM project Final Report.

[5] Tony Jenkins. 2002. On the Difficulty of Learning to Program. In *Proceedings of the 3rd Annual HEA Conference for the ICS Learning and Teaching Support Network*. 1–8.

[6] Maria Kallia and Sue Sentance. 2017. Computing Teachers' Perspectives on Threshold Concepts: Functions and Procedural Abstraction. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE '17)*. ACM, New York, NY, USA, 15–24. https://doi.org/10.1145/3137065.3137085

[7] Richard Lobb and Jenny Harlow. 2016. Coderunner: A tool for assessing computer programming skills. *ACM Inroads* 7, 1 (2016), 47–51.

[8] Davin McCall and Michael Kölling. 2014. Meaningful Categorisation of Novice Programmer Errors, In Proceedings of the 2014 IEEE Frontiers in Education (FIE) Conference. *Proceedings - Frontiers in Education Conference, FIE* 2015. https://doi.org/10.1109/FIE.2014.7044420

[9] Erik Meyer and Ray Land. 2003. Thresholds Concepts and Troublesome Knowledge: Linkages to Ways of Thinking and Practising within the Disciplines.

[10] Christian Murphy, Dan Phung, and Gail Kaiser. 2008. A distance learning approach to teaching eXtreme programming. *ACM SIGCSE Bulletin* 40, 199–203. https://doi.org/10.1145/1384271.1384325

[11] Paul Piwek, Michel Wermelinger, Robin Laney, and Richard Walker. 2019. Learning to program: from problems to code. In *Third Conference in Computing Education Practice (CEP)*. Association for Computing Machinery (ACM), Durham, United Kingdom. http://oro.open.ac.uk/58202/

[12] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 18, 1, Article 1 (Oct. 2017), 24 pages. https://doi.org/10.1145/3077618

[13] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage, London.

[14] Kate Sanders, Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Lynda Thomas, and Carol Zander. 2012. Threshold Concepts and Threshold Skills in Computing. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/2361276.2361283

[15] Kate Sanders and Robert McCartney. 2016. Threshold Concepts in Computing: Past, Present, and Future. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)*. ACM, New York, NY, USA, 91–100. https://doi.org/10.1145/2999541.2999546

[16] Simon Savage and Paul Piwek. 2019. Full report on challenges with learning to program and problem solve: an analysis of first year undergraduate Open University distance learning students' online discussions. http://oro.open.ac.uk/68073/

[17] Dermot Shinners-Kennedy and Sally A. Fincher. 2013. Identifying Threshold Concepts: From Dead End to a New Direction. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 9–18. https://doi.org/10.1145/2493394.2493396

[18] Elaine Thomas. 2019. A new approach to teaching introductory Computing and Information Technology by distance learning - addressing key issues. In *Connecting through Educational Technology - Proceedings of the European Distance and E-Learning Network 2019 Annual Conference*, Airina Volungeviciene and András Szűcs (Eds.). 292–300. http://oro.open.ac.uk/62184/